Writing with AI

AI final project – team 26

109550129 林揚哲

109550174 黃昱翰

Introduction

- Purpose
  - Generating text by trained model
- Computer is faster on heavy reading
- Inspiration
- Continue writing

Related work

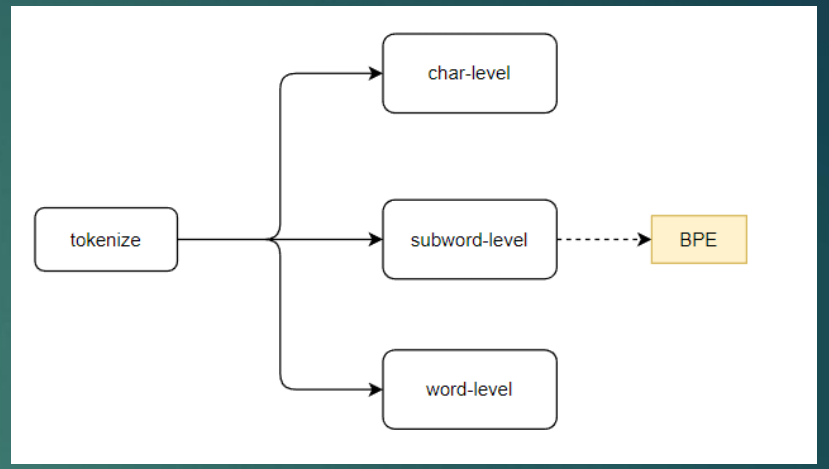
- BlinkDL/AI-Writer
  - RWKV Model
- Graykode/gpt-2-Pytorch
- huggingface/transformers
- mymuise/gpt2-quickly

Dataset

- Huyen Nguyen, SimpleBooks
  - Created from 1,573 Gutenberg books
  - 92M word-level tokens, on par with WikiText-103 (103M tokens)
  - vocabulary of 98K, a third of WikiText-103's.

Dataset – Construction

- BPE tokenizer
  - Why choose BPE tokenizer
  - Subword-level tokenizer
  - Keep mixing the highest probability characters into new subwords




```
u-n-r-e-l-a-t-e-d
u-n re-l-a-t-e-d
u-n re-l-at-e-d
u-n re-l-at-ed
un re-l-at-ed
un re-l-ated
un rel-ated
un-related
unrelated
```

Dataset – Construction (2)

- Normalizer
  - sentencepiece
  - NFKC
  - Improving operation efficiency
- Pre-tokenizer
  - Bytelevel
  - Replacing bytes of given string, Spliting into words.
- Special tokens

```
def __init__(self):
    # Initialoze tokenizer
    self.tokenizer = Tokenizer(BPE())
    # Provide the equality for unicode charactors
    # It may seen same words in different forms as equal
    self.tokenizer.normalizer = Sequence([
        # Normalization Form Compatibility Composition
        NFKC()
    ])
    self.tokenizer.pre_tokenizer = ByteLevel()
    self.tokenizer.decoder = ByteLevelDecoder()
```

```
trainer = BpeTrainer(vocab_size=50000,
    show_progress=True,
    special_tokens=[
        "<s>",
        "<pad>",
        "</s>",
```

src/tokenization.py

Dataset – Construction (3)

- Making Dataset
  - Load tokenizer
  - Encode the text
- Shuffle
  - Randomly shuffles elements of the dataset
- Batch
  - Combine elements of the dataset into batch
- Make Dataset
  - Type: tf.data.Dataset

```
# Use tokenizer to encode the data as single string
data_tokenized : str
data_pos = config.train_pos if type == "train" else config.test_pos
with open(data_pos, "r", encoding='utf-8') as f:
    tmp = f.read()
    tmp = tmp.replace("\n", " ")    # clean up the change line characters
# Add token, this can be done when having multiple input files or text
# add_special_tokoen = tokenizer.bos_token + tmp + tokenizer.eos_token
data_tokenized = tokenizer.encode(tmp)
print("\tText Encoded...")

# Slice data with equal quantity
examples, inputs, labels = [], [], []
for i in range(0, len(data_tokenized) - config.block_size + 1, config.block_
    examples.append(data_tokenized[i:i + config.block_size])
for ex in examples:
    inputs.append(ex[:-1])
    labels.append(ex[1:])

dataset = tf.data.Dataset.from_tensor_slices((inputs, labels))
# Shuffles & Batch
dataset = dataset.shuffle(buffer_size=config.buffer_size) \
    .batch(config.batch_size, drop_remainder=True)
return dataset
```

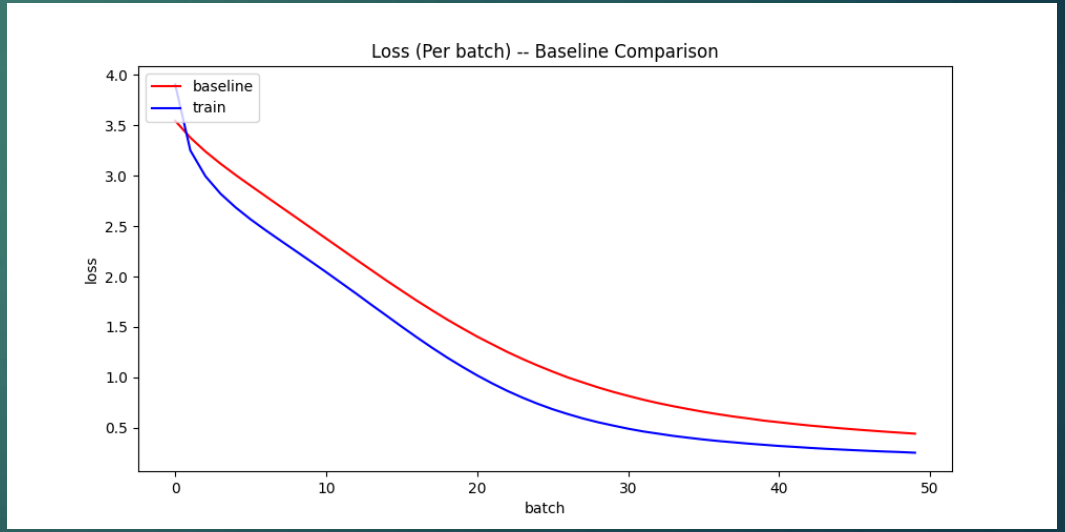
train.py – making dataset

Baselines

- Comparison with other's program
  - mymuise/gpt2-quickly
- mymuise/gpt2-quickly
  - WordPiece Tokenizer
  - Without normalizer

```python
def main():
    tokenizer = BertWordPieceTokenizer()
    tokenizer.train(files=[configs.data.raw],
                    vocab_size=52000, min_frequency=1)
    tokenizer.save_model(configs.data.path)
    print(f"save to {configs.data.path}")
    # GPT2Tokenizer.from_pretrained()
```

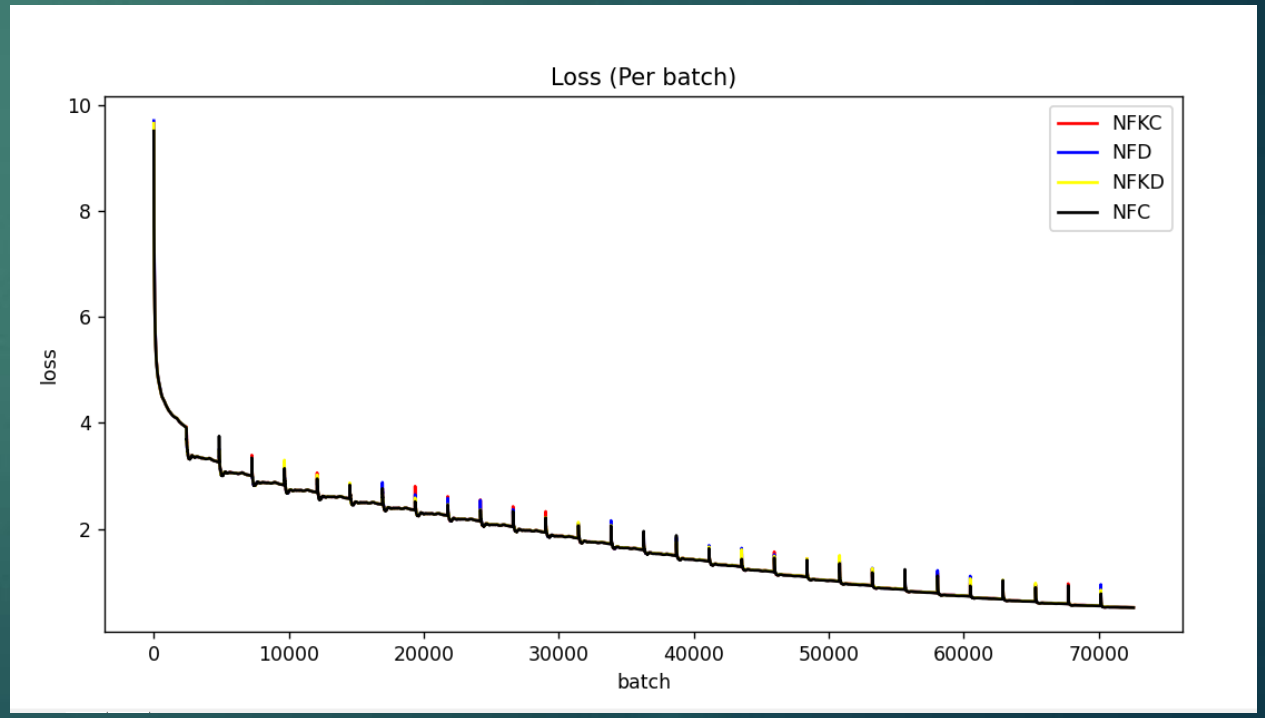
gpt2-quickly: build_tokenizer.py



Comparison baselines & trained model

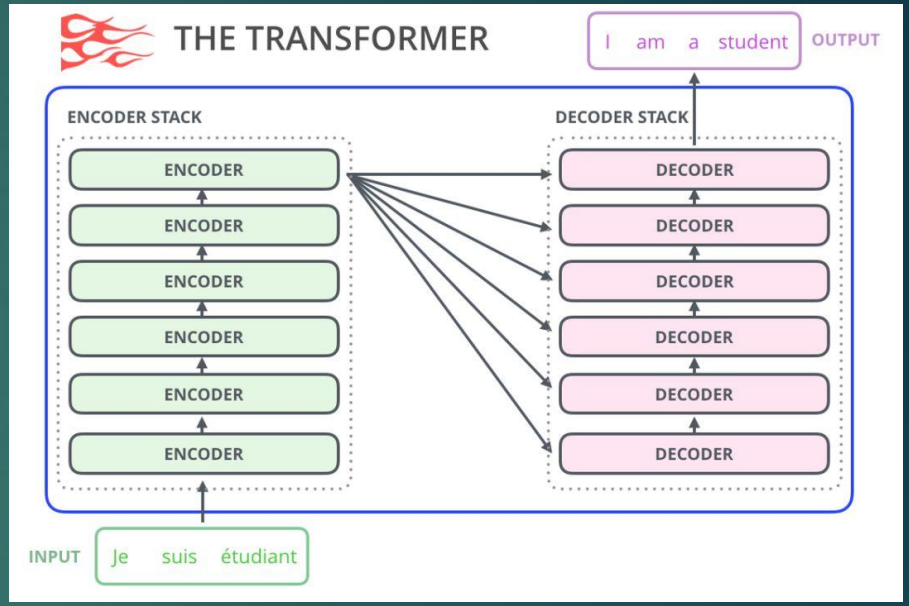
Baselines

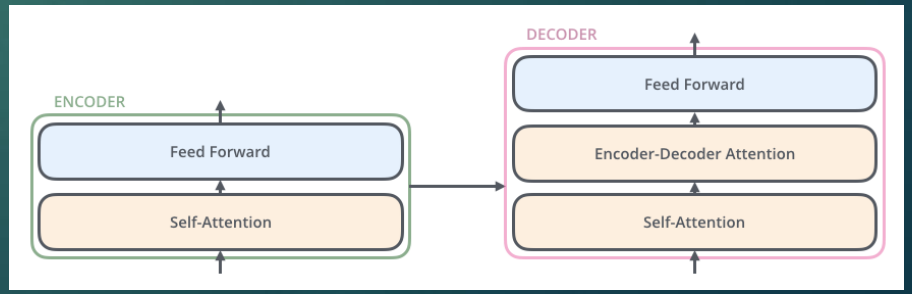
- Comparison with other normalizers
  - No noticeable difference
- Different Normalization Form
  - NFD
  - NFC
  - NFKC
  - NFKD



Main Approach

- GPT-2
  - Generative Pre-trained Transformer
  - BPE Encoding
  - Transformer
  - Auto-regression
  - Self-Attention

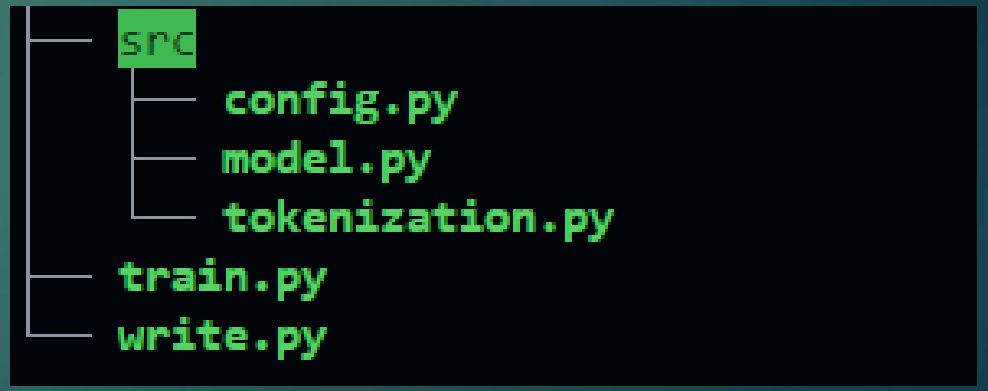




$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Main Approach – Code Structure

- src/*
  - config.py: Program configuration
  - model.py: Model definition
  - tokenization.py: Build BPE tokenizer.
- train.py
  - Main procedure of training
- write.py
  - Generating text via pre-trained model
- Procedure
  - tokenization.py -> train.py



Code structure

Main Approach

- Byte-Pair Encoding
  - Using tokenizer built in former
  - translate text into numeric indices
- Construct the Model
  - Provided dataset
  - Callback function
- Text-generation
  - Top-k sampling
  - Redistributed on top K words

```
## Init Model       ##
print("==> Init Model:")
train_model = TextModel(config=config, tokenizer=BPE_tokenizer, model_name="train")
test_model = TextModel(config=config, tokenizer=test_BPE_tokenizer, model_name="test")
print("\tModel initialized...")
```

Model initialization

```
text = "Did you hear that ?"
tokenizer = BPE_tokenizer
text_generator = TextGenerationPipeline(train_model.model, tokenizer)
print (f'''Result:
    {text_generator(
        text_inputs=text,
        max_length=128,
        do_sample=True,
        top_k=10,
        eos_token_id=tokenizer.get_vocab().get("</s>", 0)
    )[0]['generated_text']}
    '''
)
```

Text generation with top-k sampling

Evaluation Metric

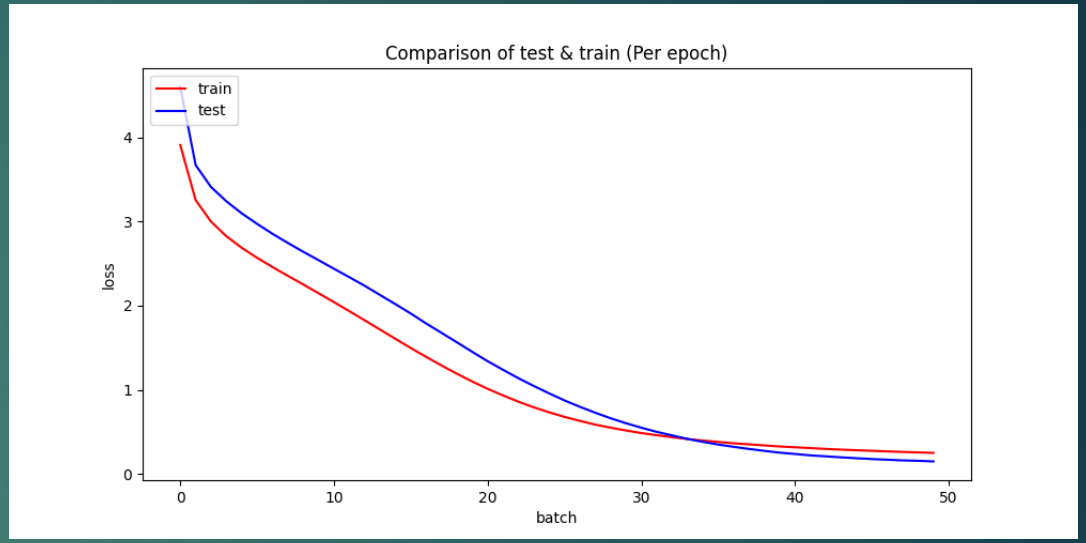
- Loss function
  - Sparse Categorical Cross Entropy
- Metric
  - Accuracy
- Optimizer
  - We choose Adam optimizer
  - Compared with
    - SGD
    - Momentum
    - AdaGrad

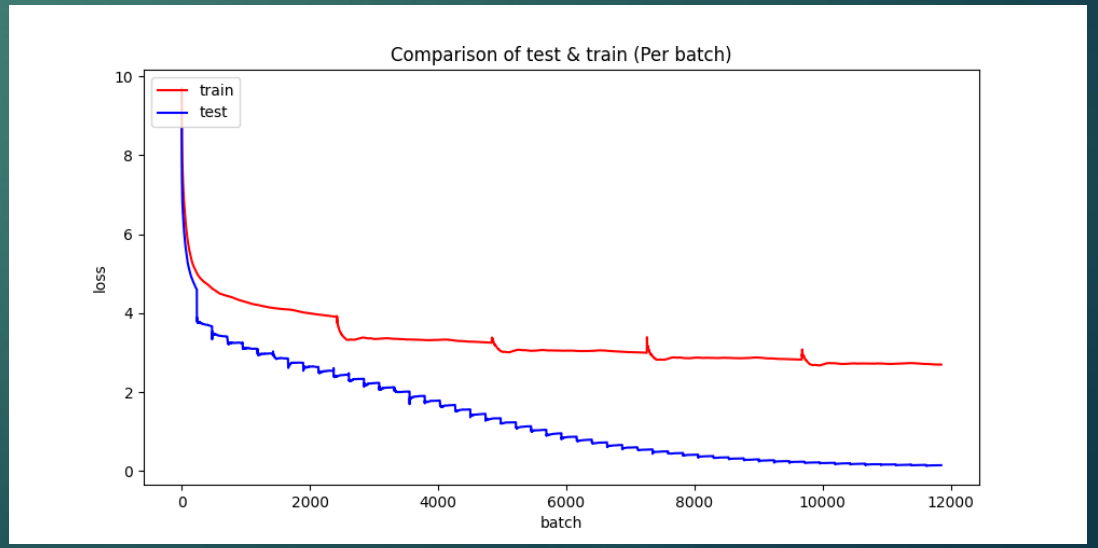
```
# Define
self.optimizer = tf.keras.optimizers.Adam(learning_rate=5e-5, epsilon=1e-08, clipnorm=1.0)
self.loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
self.metric = tf.keras.metrics.SparseCategoricalAccuracy('accuracy')
```

Defined in model.py, Init of TextModel

Evaluation Metric – Train/Test Split

- Separate train, test dataset
  - Validate model hyper-parameters
  - Test Dataset
    - About 10% amount of Train Dataset
    - No cross data

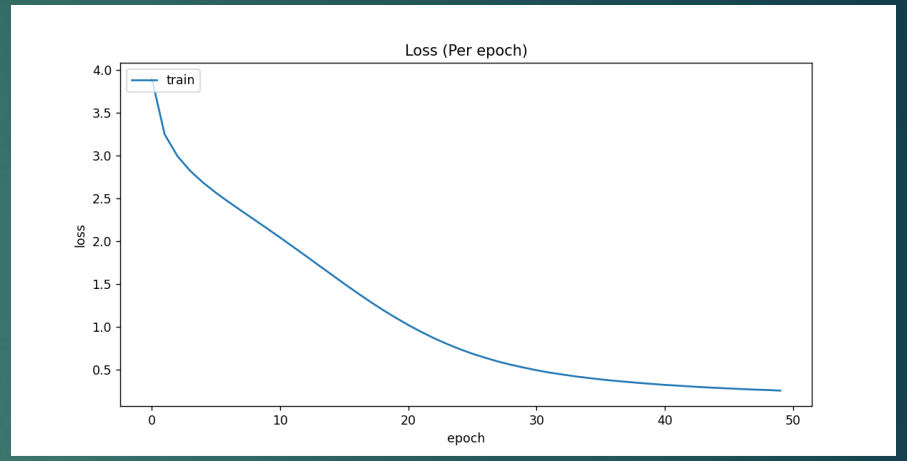


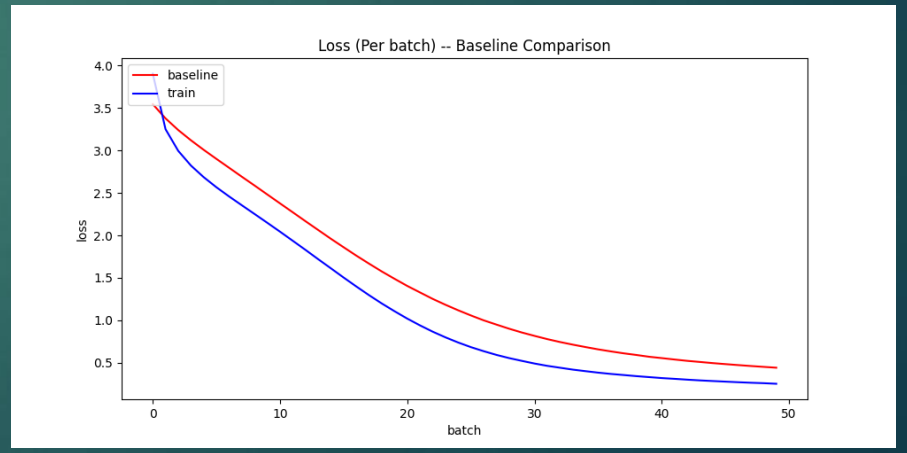


Result & Analysis

- Baseline
  - Modified from mymuise/gpt2-quickly
- Text Generation
  - Cut on the max length
  - Stable character name

```
        Training result output....
Input: Did you hear that
Setting `pad_token_id` to 2 (first `eos_token_id`) to generate sequence
Result:
                Did you hear that  " I believe the gypsies are trying to find ou
t about this thing, " said Bert. " If they did they 'd have gone away. However,
I 'll take you home with me for a little while, and then let my dog go away with
 us. "  " Perhaps the gypsies did, " said Mr. Bobbsey. " They took Helen's doll
and were camping on the island of Lakeport. "  " I 'll ask about that when they'
ve camped on Blueberry Island, " said Mr. Bobbsey.  " I 'Cause though they did n
ot let the gypsies know
```





Future Work

- Fixed / Optimized Problem
  - Preprocess
  - More dataset
  - Different Language
  - Collect general sentence ending

Thanks for your listening !